Neural Computation (VS 265), Problem Set 4

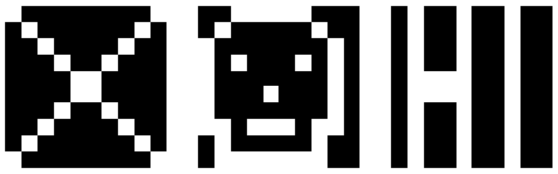
Due date: October 25, 3:30pm

Fall 2022

General guidelines:

- We are grading problem sets anonymously. **Include your student ID in the submission, but do not include your name.**
- You may work in small groups of 2-3. Note that you are responsible for writing up and submitting your submission individually.
- You are expected to attach any code you used for this assignment but will be evaluated primarily on the writeup.
- If you are including animations as part of your submission, please attach these files separately in your submission (it should not be necessary to run any code to view your animations).

Part 1: Hopfield Network



i. Implement the discrete Hopfield (1982) network model and store the three simple 10×10 image patterns above (they can be found on the course website) in the weight matrix. Next, corrupt one of the images by randomly flipping some percentage of the binary pixels. Feed the corrupted version as input to the network, and run the dynamics until equilibrium. Make a plot showing side by side the initial corrupted image, an intermediate image between the network's initial and final state, and the final image once the network has converged. For small amounts of corruption, it should converge to the same image. Try corrupting the image to the point where the network no longer converges to the correct image, and make a similar plot to the previous one (initial, intermediate, and final state of the network).

ii. The *capacity* of a Hopfield network refers to the maximum number of patterns that can be stored and successfully retrieved from the network. To explore the capacity, create datasets containing different numbers of random patterns and store them in the weight matrix. To evaluate the model's retrieval accuracy, initialize the network state to one of the stored patterns, let it run to equilibrium, and then compute the Hamming distance between the final state and the initial state. Repeat this process multiple times for each dataset and plot the mean percentage of errors as a function of the number of stored patterns. You will need to choose the number of stored patters for each dataset so that your plot traces out the regime from error-free retrieval to 50% bit errors.

iii. The example above (i) used the Hopfield network to store images. However, the Hopfield network can store any type of pattern. What other types of patterns might the brain want to store that a Hopfield network could facilitate?

Part 2: Ring Attractor Network

The dynamics of Kechen Zhang's (1996) ring attractor model are defined as:

$$\tau \frac{du}{dt} = -u + w * \sigma(u)$$

where $u = u(\theta, t)$ is the synaptic input current, $w = w(\theta, t)$ defines weights between units, $\sigma(u) = f$ is the firing rate of the units, and $*$ denotes convolution. (See the paper for additional details and explanation.) For this part, you will implement the dynamics of this network.

i. First, assume that $\sigma(x)$ is defined as $a \ln^{\beta}(1 + e^{b(x+c)})$, with parameters $\beta = 0.8, b = 10, c = 0.5$, and $a = 6.34$. The next step is to find a set of weights w such that the network stabilizes to a single bump of activity. This can be accomplished by using the following equation:

$$\hat{w}_n = \frac{\hat{u}_n \hat{f}_n}{\lambda + |\hat{f}_n|^2}$$

where here $u = \ \sigma^{-1}$ (the inverse function of σ), and $\hat{w}_n$, $\hat{f}_n$, and $\hat{u}_n$ are the n-th Fourier coefficients of w, f, and u respectively.

Try varying λ in the range: $\lambda = \{10^{-1}, 10^{-2}, \ldots, 10^{-5}\} \times \max|\hat{f}_n|^2$. How does varying λ affect your weights? (And why does this effect occur?)

ii. Verify that your solution works by initializing the values of u randomly and show that the dynamics converge to a stabilized bump. Save an animation or show multiple timesteps of these updating dynamics for a few different random initializations. (Hint: see Figure 3 of the paper for the type of results you should expect to get.)

iii. Next, add a shifting mechanism so that the bump position shifts over time. You can accomplish this by adding a small term to w proportional to the derivative of the weights. For further details, please visit section 5 of the paper. Again, save an animation or show multiple timesteps of your result.